Universidad de Deusto
ESIDE

AJAX: Asynchronous JavaScript and XML (AJAX)

Dr. Diego Lz. de Ipiña Gz. de Artaza
http://paginaspersonales.deusto.es/dipina
dipina@eside.deusto.es

Universidad de Deusto
ESIDE

Contenidos

- Introducción a AJAX
- Análisis de sus componentes:
  - XHTML y CSS
  - JavaScript
  - DOM y XSLT
- Hola mundo en AJAX
- Frameworks AJAX
- Ejemplos avanzados
- Conclusión

Concepto AJAX

- AJAX (Asynchronous Javascript and XML) es una técnica de desarrollo web que genera aplicaciones web interactivas combinando:
  - XHTML y CSS para la presentación de información
  - Document Object Model (DOM) para visualizar dinámicamente e interactuar con la información presentada
  - XML, XSLT para intercambiar y manipular datos
    - JSON y JSON-RPC pueden ser alternativas a XML/XSLT
  - `XMLHttpRequest` para recuperar datos asíncronamente
  - Javascript como nexo de unión de todas estas tecnologías
- AJAX es un **patrón de diseño** que propone un nuevo modelo de interacción Web combinando las tecnologías anteriores
- Los navegadores que soportan las tecnologías mencionadas son las plataformas en las que se ejecutan las aplicaciones AJAX (Firefox, Iexplorer, Opera, Konqueror y Safari)

Universidad de Deusto
ESIDE

¿Por qué AJAX?

- Las aplicaciones web proliferan debido a su simplicidad, pero:
  - Ofrecen una menor interactividad y usabilidad en comparación con las aplicaciones desktop.
  - La interacción del usuario con una aplicación web se interrumpe cada vez que se necesita algo del servidor
- Varias tecnologías han sido diseñadas para resolver este problema:
  - Java Applets, FLASH
- ¡AJAX es una nueva solución!
  - Término acuñado por Jesse James Garrett en http://www.adaptivepath.com/publications/essays/archives/000385.php
  - No requiere plugins o capacidades específicas de ciertos navegadores.
  - ¿En competición con Smart Clients de MicroSoft?

Universidad de Deusto
ESIDE

DTML y AJAX, ¿lo mismo?

- DTML (Dynamic HTML) técnica de desarrollo de webs interactivas combinando HTML, client-side scripting con JavaScript y el lenguaje de definición de estilos (CSS)
  - Realmente AJAX es un refinamiento de DTML
- Al igual que en AJAX el principal problema de DTML es el diferente soporte de estas tecnologías en los navegadores y los diferentes tamaños de pantalla de usuarios
  - El mismo DOM (Document Object Model) es soportado por las últimas versiones de los navegadores

Universidad de Deusto
ESIDE

Características AJAX

- Las aplicaciones son más interactivas, responden a las interacciones del usuario más rápidamente, al estilo desktop
- Estas aplicaciones tienen un *look and feel* muy similar a las aplicaciones de sobremesa tradicionales sin depender de plug-ins o características específicas de los navegadores
- Se reduce el tamaño de la información intercambiada
  - Muchas micro-peticiones, pero el flujo de datos global es inferior
- Se libera de procesamiento a la parte servidora (se realiza en la parte cliente)
- AJAX actualiza porciones de la página en vez de la página completa
- Es necesario asegurarse que una aplicación AJAX funciona en todo navegador, lo mismo que en DHTML

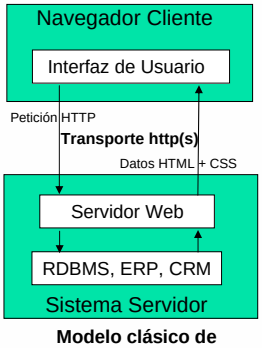
Universidad de Deusto
ESIDE

AJAX vs. Aplicaciones Web Tradicionales

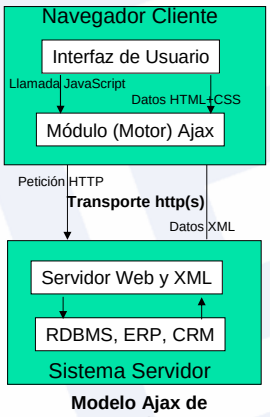
- Las aplicaciones web tradicionales:
  - Requieren que el usuario rellene formularios y haga un POST de los mismos a un servidor web
  - El servidor web procesa los datos enviados y genera un nueva página web en consecuencia
    - Mientras el servidor realiza cierto procesamiento – recupera datos, realiza cálculos – para generar una página, el usuario ha de esperar
  - Aparte de los datos de la aplicación la información de formateo en XHTML debe ser transmitida
    - En ocasiones el HTML de una página a la siguiente varía muy ligeramente.
- AJAX solamente envía peticiones a los servidores vía HTTP y procesa la información recuperada (únicamente la necesitada) mediante JavaScript

Universidad de Deusto
ESIDE

AJAX vs. Aplicaciones Web Tradicionales



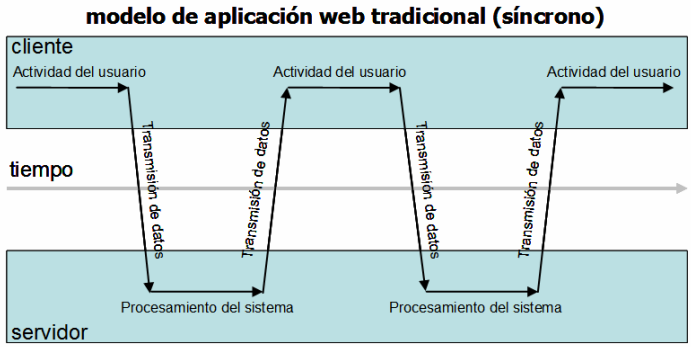
Modelo clásico de aplicación web



Modelo Ajax de aplicación web

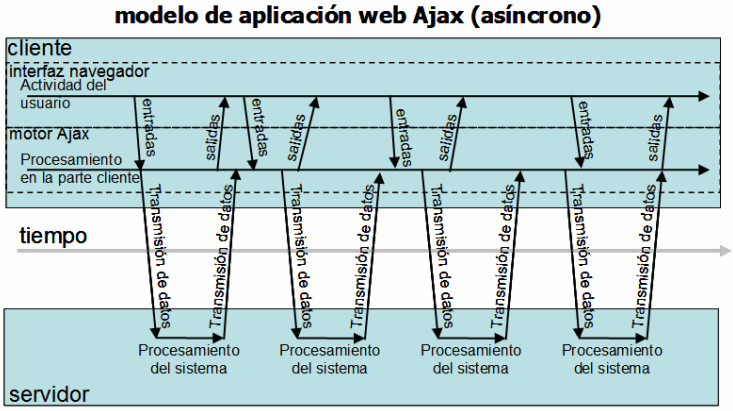
Universidad de Deusto
ESIDE

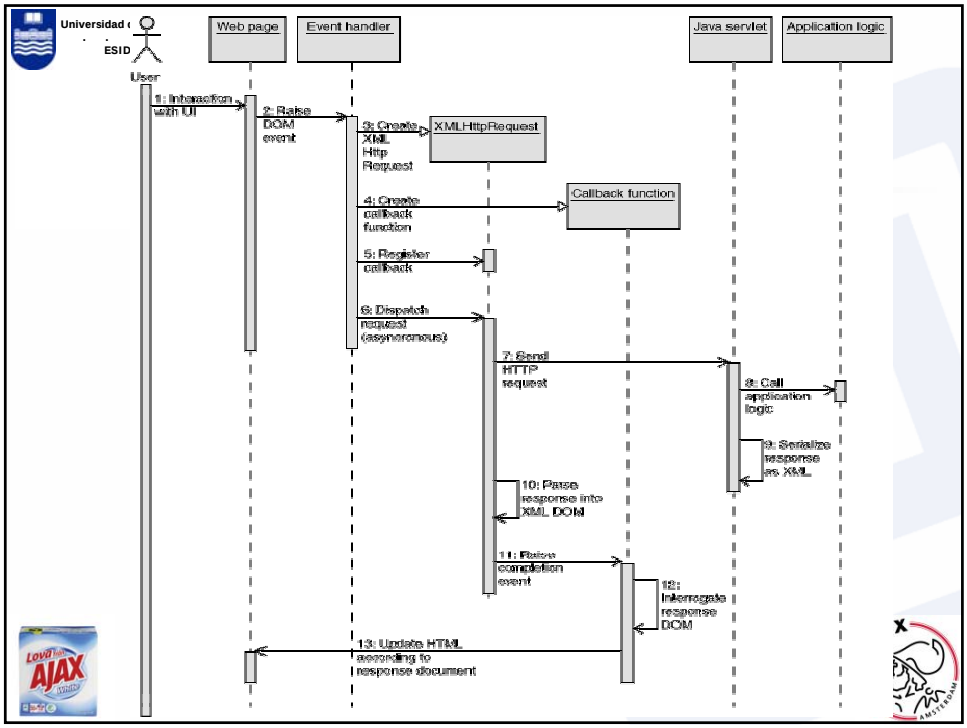
AJAX vs. Aplicaciones Web Tradicionales



Universidad de Deusto
ESIDE

AJAX vs. Aplicaciones Web Tradicionales





Universidad de Deusto
ESIDE

El motor AJAX

- El motor AJAX (AJAX engine) colocado entre el usuario y el servidor web evita el ciclo *start-stop-start-stop* característico de las aplicaciones web tradicionales
  - No es mas que un fichero JavaScript que acompaña al HTML
- Este motor es cargado al inicio de la sesión y tiene una doble responsabilidad:
  - Generar la interfaz visualizada por el usuario
  - Comunicarse con el servidor en representación del usuario
    - Esta **interacción** ocurre de manera **asíncrona** evitando que el usuario vea una página blanca o el reloj de arena (de espera) cada vez que realice una acción.
- Cada interacción del usuario revierte en una invocación al AJAX engine que puede ser resuelta local (validación de datos) o remotamente (objeto `XMLHttpRequest`).

Universidad de Deusto
ESIDE

Aplicaciones AJAX famosas

- Empresas de renombre internacional están definiendo soluciones basadas en AJAX:
  - Google
    - Orkut (https://www.orkut.com/Login.aspx) es una comunidad virtual que conecta online a gente a través de una red de amigos.
    - Gmail (www.gmail.com)
    - Google Suggest (http://www.google.com/webhp?complete=1&hl=en) → sugiere valores de búsqueda a medida que escribes caracteres
    - Google Maps (http://maps.google.com/)
  - Yahoo!
    - Flickr (http://www.flickr.com/) es una aplicación para gestionar y compartir fotos
    - Oddpost (http://oddpost.com/learnmore)
      - El equipo de webmail Oddpost actualmente están trabajando en rediseñar Yahoo! Mail siguiendo la filosofía AJAX
- En definitiva, AJAX no es solamente un buena solución técnica sino que además tiene una gran aplicabilidad como se demuestra por aplicaciones reales complejas.

Universidad de Deusto
ESIDE

Google Gmail (www.gmail.com)

- Gmail coloca un motor UI en la parte cliente en forma de un módulo/fichero JavaScript
- Cuando hace login en Gmail una copia de ese motor es descargado en uno de los HTML page frames y permanece ahí por el resto de la sesión
- Este enfoque no es nuevo, fue adoptado por Oddpost
- Las acciones sobre la interfaz de Gmail son ruteadas a través del motor UI en el navegador, que realiza peticiones HTTP (con el objeto `XmlHttpRequest`) al servidor de Gmail
- El servidor Gmail envía información en `DataPack` que son interpretados por el motor UI para actualizar dinámicamente la interfaz
  - DataPack es similar a Javascript Object Notation (JSON – www.json.org)
- Una razón del éxito de Gmail es su mucho mayor interactividad y usabilidad.
  - Otros servicios de correo como Hotmail y Yahoo! Mail siguen un enfoque tradicional de desarrollo web y vuelven a descargar toda la interfaz completamente después de cada acción.

Universidad de Deusto
ESIDE

Presentación en AJAX: XHTML y CSS

- XHTML define la estructura de un documento
- CSS proporciona el “look and feel” al documento

Universidad de Deusto
ESIDE

XHTML (I)

- XHTML (eXtensible HyperText Mark-up Language) es la siguiente especificación de HTML.
- Es HTML ***basado en XML***.
- Los usuarios pueden añadir sus propias etiquetas al contenido.
- XHTML 1.1 está modularizado: sus etiquetas se agrupan en módulos, de esta forma, todo el lenguaje se encuentra descompuesto en módulos.
- El significado de las etiquetas y su uso sigue el estándar HTML 4.01.

Universidad de Deusto
ESIDE

XHTML (II)

- Algunos de estos módulos son:
  - Estructura
  - Texto
  - Hipertexto
  - Lista
  - Applet
  - Formularios básicos
  - Formularios
  - Imagen
  - Marcos
  - ...
- Además se pueden añadir nuevos módulos.

Universidad de Deusto
ESIDE

XHTML (III)

- XHTML Basic es una versión de XHTML adecuada para terminales móviles.
- Incluye los módulos:
  - Estructura (`body, head, html, title`)
  - Texto (`abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var`)
  - Hipertexto (`a`)

Universidad de Deusto
ESIDE

XHTML (IV)

- Incluye los módulos (cont.):
  - Lista (`dl, dt, dd, ol, ul, li`)
  - Formularios básicos (`form, input, label, select, option, textarea`)
  - Tablas básicas (`caption, table, td, th, tr`)
  - Imagen (`img`)
  - Objeto (`object, param`)
  - Metainformación (`meta`)
  - Link (`link`)
  - Base (`base`)

Universidad de Deusto
ESIDE

Cascading Style Sheets (CSS)

- CSS es un lenguaje de estilos utilizado para describir la presentación de un documento escrito en lenguaje de marcado.
- Su uso más típico es proporcionar el “look and feel” de páginas web en XHTML, aunque puede ser utilizado en aplicaciones XML como SVG o XUL
- La especificación de CSS es mantenida por el World Wide Web Consortium (W3C)
- A través de CSS se pueden definir colores, fuentes, layouts y otros aspectos de la presentación de un documento.
- Sirve para separar la estructura de un documento (HTML) de su presentación (CSS).
  - http://en.wikipedia.org/wiki/Cascading_Style_Sheets

Universidad de Deusto
ESIDE

Utilidad de CSS

- Antes de CSS, los autores de páginas HTML utilizaban elementos como `<font>` o `<b>` para cambiar el formato de las páginas, combinaban estructura y presentación
- Por ejemplo, si queremos que una cabecera esté centrada, en cursiva, en rojo y con la fuente Times New Roman, podríamos hacer:

```
<h2 align="center"><font color="red" size="+4" face="Times
   New Roman, serif"><em>Usage of CSS</em></font></h2>
```

- Con CSS, el elemento `h2` puede ser utilizado para dar estructura al texto, mientras que la hoja de estilo le da sus características presentacionales. Así en XHTML tendríamos:

```
<h2>Usage of CSS</h2>
```

- Mientras que en CSS definiríamos las características de formato para todas las cabeceras `h2`:

```
h2 {
    text-align: center;
    color: red;
    font: italic large "Times New Roman", serif;
}
```

Universidad de Deusto
ESIDE

CSS

- La información de un CSS puede ser provista por:
  - Autor de estilos
    - Un fichero CSS referenciado desde el documento
    - Embebido en el documento
  - Estilo del usuario
    - Un fichero CSS local es especificado por el usuario en su navegador para ser aplicado a todos los documentos
  - Estilo del User-Agent
    - La hoja de estilo por defecto aplicada por el navegador
- Las principales ventajas de CSS son:
  - Información de presentación para una página web completa o una colección de páginas residente en un único lugar donde puede ser actualizada
  - Diferentes usuarios pueden tener sus propias hojas de estilo (fuentes grandes)
  - El código del documento es reducido en tamaño y complejidad, porque no tiene que contener información de presentación

CSS

- Los elementos de un documento son seleccionados a través de selectores:
  - Todos los elementos → '*'
  - Por nombre de elemento → 'p' ó 'h2'
  - Descendentes
    - Todos los elementos a descendientes de `li` '`li a`'
  - Atributos de clase o identificadores
    - *`.nombreclase`* y/o *`#id`* para elementos con `class="`*`nombreclase`*`"` or `id="`*`id`*`"`
  - Estilos adyacentes
    - Todos los elementos A precedidos por h2 → '`h2+a`'
- Existen también pseudoclases que redefinen el comportamiento sobre elementos:
  - `hover` → define un estilo cuando el usuario apunta a un elemento
    - `a:hover` o `#elementid:hover`
  - `:first-line`, `:visited` or `:before`.

Universidad de Deusto
ESIDE

CSS

- Una hoja de estilos contiene un conjunto de reglas
- Cada regla consiste de un selector y un bloque declarativo
- Un bloque declarativo contiene un conjunto de declaraciones separadas por ;
- Cada declaración consiste de una propiedad, : y un valor

Ejemplo CSS

```
p {
   font-family: "Garamond", serif;
}
h2 {
   font-size: 110%;
   color: red;
   background: white;
}
.note {
   color: red;
   background: yellow;
   font-weight: bold;
}
p#paragraph1 {
   margin: none;
}
a:hover {
   text-decoration: none;
}
```

Explicación ejemplo CSS

- Hay 5 reglas, con los selectores `p`, `h2`, `.note`, `p#paragraph1` y `a:hover`.
- Un ejemplo de declaración es `color: red`, que asigna el valor rojo a una propiedad.
- Las dos primeras reglas definen estilos para los elementos HTML `p` (párrafo) y `h2` (cabecera de segundo nivel)
  - El elemento párrafo se renderizará en la fuente `Garamond` y si no está disponible en `Serif`.
  - La cabecera será renderizada en rojo sobre fondo blanco.
- La tercera regla define una clase en CSS que puede ser asignada a cualquier elemento del documento, mediante el atributo `class`. Por ejemplo:

```
<p class="note">This paragraph will be rendered in red and
   bold, with a yellow background.</p>
```

- La cuarta regla afectará al elemento `p` cuyo atributo `id` tenga el valor `paragraph1`. No tendrá márgenes dentro de su caja de renderización
- La última regla define la acción `hover` para todos los elementos `<a>`.
  - Quitará el subrayado cuando el ratón se mueve sobre el elemento enlace.
- Los comentarios en CSS se indican con: `/* comment */`

Universidad de Deusto
ESIDE

Lógica en AJAX: JavaScript

- Es un lenguaje de scripting orientado a objetos
- Es utilizado especialmente para permitir la programación en navegadores
  - Algunos lo denominan como “webpage enhancement language”
  - Otros dicen que es un “client-side scripting language”
- La versión estándar de JavaScript, denominada ECMAScript va por la versión 1.5
- Java y Javascript son dos lenguajes totalmente diferentes
  - Presentan similitudes sintácticas, están basados en C
  - Semánticamente son muy diferentes
- Para escribir scripts web en JavaScript hay que familiarizarse con:
  - El lenguaje JavaScript
  - El modelo de objetos del documento (DOM) de XHTML

Universidad de Deusto
ESIDE

Usos de JavaScript

- El motor de JavaScript reside en un programa huésped y puede operar sobre su entorno
- El JavaScript de un página puede manipular y recuperar su contenido a través de la interfaz DOM
  - El mayor uso de JavaScript es para crear aplicaciones web dinámicas
    - Interactuando con el DOM de una página HTML, JavaScript puede:
      - Abrir nuevas ventanas
      - Cambiar imágenes cuando el ratón se mueve encima de ellas
      - Validar campos de entrada, etc.
  - JavaScript también es usado por:
    - Adobe Acrobat
    - La plataforma Mozilla con XUL

Universidad de Deusto
ESIDE

Sintaxis JavaScript

- Es case-sensitive
- El "`;`" no es necesario al final de cada sentencia
- Los comentarios siguen la misma sintáxis que en C++, `/**/` y `//`
- Las variables se declaran dinámicamente asignándoles un valor o alternativamente precediéndolas con la palabra `var`
  - Las variables declaradas con `var` dentro de una función son locales

Universidad de Deusto
ESIDE

OOP en JavaScript

- Los objetos en Javascript toman la forma internamente de un array asociativo, mapean nombres de propiedades a valores
  - Tiene varios tipos de objetos ya definidos:
    - `Array`, `Boolean`, `Date`, `Function`, `Math`, `Number`, `Object`, `RegExp` y `String`
    - También existen objetos soportados por el entorno de ejecución como `window`, `form`, `links` en el DOM.
- JavaScript es un lenguaje orientado a objetos basado en prototipos.
  - La herencia tiene lugar entre objetos en vez de clases
  - Se pueden añadir nuevas propiedades y métodos a un objeto a través de la palabra clave `prototype`
- No es obligatorio borrar objetos ya que tiene garbage collector
- JavaScript tiene los siguientes tipos de datos primitivos:
  - Números → 4, 4.4, 44
  - Strings → `'ab'` o `"ab"`
  - Objetos → conjunto de strings y numbers

Universidad de Deusto
ESIDE

Ejemplo OOP en JavaScript

```
// constructor function
function MyObject(attributeA, attributeB) {
  this.attributeA = attributeA
  this.attributeB = attributeB
}

// create an Object
obj = new MyObject('red', 1000);

// access an attribute of obj
alert(obj.attributeA);

// access an attribute with the associative array notation
alert(obj["attributeA"]);

// add an new attribute
obj.attributeC = new Date();

// remove an attribute of obj
delete obj.attributeB;

// remove the whole Object
delete obj;
```

Universidad de Deusto
ESIDE

Ejemplo Herencia

```
function Base() {
  this.Override = function() {
    alert("Base::Override()");
  }
  this.BaseFunction = function() {
    alert("Base::BaseFunction()");
  }
}
function Derive()
{
   this.Override = function() {
     alert("Derive::Override()");
   }
}
Derive.prototype = new Base();

d = new Derive();
d.Override();
d.BaseFunction();
d.__proto__.Override(); // mozilla only
```

- produce:

```
Derive::Override()
Base::BaseFunction()
Base::Override()  // mozilla only
```

Universidad de Deusto
ESIDE

Acceso público y privado a variables en JavaScript

```
// declaration
function myClass() {
  // variables
  var privateVar = "I am private";
  this.publicVar = "I am public";

  // functions
  var privateFunction = function() {
    alert("I am private");
  }
  this.publicFunction = function(){
    alert("I am public");
  }
}

// usage
var myInstance = new myClass();

myInstance.publicVar = "new value";
alert(myInstance.privateVar);            // undefined!

myInstance.publicFunction();
myInstance.privateFunction();            // error!
```

Universidad de Deusto
ESIDE

Estructuras de Datos

- Arrays → mapas de enteros a objetos
  - Define métodos como `join`, `slice` y `push`
  - Tiene la propiedad `length`
  - Ejemplo:

```
myArray = [0,1,,,4,5]; // array with length
  6 and 4 elements
myArray = new Array(0,1,2,3,4,5); // array
  with length 6 and 6 elements
myArray = new Array(365); // an empty array
  with length 365
```

Estructuras de Control

- Son idénticas a las que nos encontramos en C++
- if … else

```
if (theForm.mail.value == "" || /^\w+([\.-
  ]?\w+)*@\w+([\.-
  ]?\w+)*(\.\w{2,3})+$/.test(theForm.mail.value)) {
   alert("++ Invalid E-mail address! Please re-
  enter. ++");
   return false;
}
```

- Operador ?

```
condition ? statement : statement;
```

- Bucle while:

```
while (condition) { statements }
```

- Do ... while

```
do { statements } while (condition);
```

Universidad de Deusto
ESIDE

Estructuras de Control

- For loop

```
for ([initial-expression]; [condition]; [increment-
  expression]) { statements }
```

- For in loop

```
for (variable in object) { statements }
```

- Switch

```
switch (expression) {
   case label1 :
      statements;
      break;
   case label2 :
      statements;
      break;
   default :
      statements;
}
```

Funciones

- Una función es un bloque de instrucciones a las que se ha dado un nombre
- Los argumentos de un función pueden ser accedidos a través de la palabra clave `arguments`
- No es necesario invocar a una función con todos los parámetros declarados, los que no se pasan toman el valor `undefined`
- Ejemplo:

```
function gcd(segmentA, segmentB) {
   while(segmentA!=segmentB)
      if(segmentA>segmentB)
         segmentA -= segmentB;
      else
         segmentB -= segmentA;
      return(segmentA);
}
```

Universidad de Deusto
ESIDE

Manejo de Errores

- A través de las típicas construcciones `try/catch/finally`
- Ejemplo:

```
try {
   // Create an array
   arr = new Array();
   // Call a function that may not succeed
   func(arr);
} catch (...) {
   // Recover from error
   logError();
} finally {
   // Even if a fatal error occurred, we can still
  free our array
   delete arr;
}
```

- La función `eval()` permite evaluar código JavaScript en tiempo de ejecución.

Universidad de Deusto
ESIDE

Añadiendo JavaScript a HMTL

- Para incluir un fragmento de JavaScript en una página HTML, el código se encierra entre las etiquetas `<script>` y `</script>`,normalmente colocadas en la sección `<head>`

```
<script language="JavaScript"
        type="text/javascript ">
    // tu código
</script>
```

- También se puede incluir un fichero externo de JavaScript con:

```
<script language='JavaScript'
  src='MyFunctions.js'/>
```

- Además desde cualquier elemento se puede invocar a JavaScript:

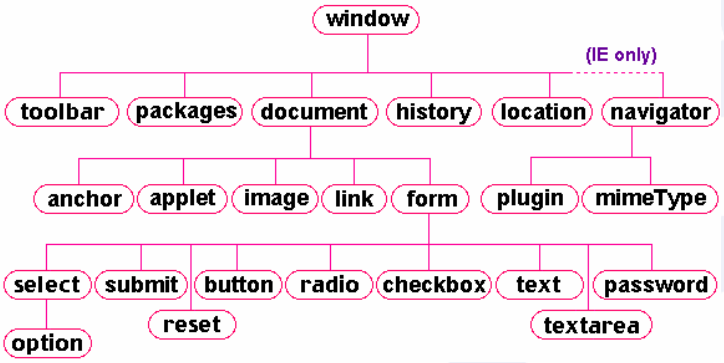
```
<a href="javascript:history.back()">go back</a>
<a href="javascript:history.go(-3)">go back</a>
```

Universidad de Deusto
ESIDE

El modelo de objetos HTML DOM utilizado por JavaScript

- Programar en JavaScript supone operar sobre el modelo de objetos del documento (DOM)
  - `window` → el objeto central que representa el navegador, contiene los objetos accesible a través de propiedades:
    - Location (`window.location`)
    - Document (`window.document`)
    - History (`window.history`)
  - Además, contiene una serie of métodos como `alert()`, `confirm()` y `prompt()`
  - No es necesario mencionar este objeto, implícitamente se asume que se invoca algo sobre él
- En la siguiente URL tenemos una excelente documentación de todos los elementos permitidos en el DOM de un documento XHTML:
  - http://www.w3schools.com/htmldom/default.asp

Operando sobre el documento en AJAX: JavaScript DOM



Manejo de eventos

- JavaScript añade interactividad a tus páginas web permitiendo escribir código dirigido por eventos
- Ejemplo: conectar un botón a una función que es invocada cuando el usuario hace clic:

```
<input type=button name=send
  onclick="handler">
```

- Eventos definidos por algunos controles HTML:
  - Button → `onBlur`, `onClick`, `onFocus`
  - Form → `onReset`, `onSubmit`
  - Frame → `onLoad`, `onUnload`
  - Link → `onClick`, `onMouseOut`, `onMouseOver`
  - Select → `onBlur`, `onChange`, `onFocus`
  - Text → `onBlur`, `onChange`, `onFocus`
  - Window → `onBlur`, `onError`, `onFocus`, `onLoad`, `onUnload`

Ejemplo Manejo de Eventos

```
<html>
<head>
<script language="javascript" type="text/javascript">
<!--
function clearBox(box) {
   if(box.value==box.defaultValue) {
        box.value = "";
   }

}
-->
</script>
</head>
<body>
<form>
<input type="text" id="email" name="email" value="enter email address"
   onFocus="clearBox(this);">
<input type="submit" name="Submit" value="Subscribe">
</form>
</body>
</html>
```

Objeto `Window`

- Representa la ventana del navegador, es la raíz de la jerarquía de objetos de JavaScript
- Propiedades:
  - `closed`
  - `defaultStatus`
  - `document`
  - `frames[]`
  - `location`
- Métodos:
  - `alert()`
  - `confirm()`
  - `prompt()`
  - `open()`
  - `setTimeout()`

Universidad de Deusto
ESIDE

Objeto `String`

- Propiedad `length` devuelve su longitud
- Métodos:
  - `charAt()`
  - `indexOf()`
  - `lastIndexOf()`
  - `split()`
  - `substring()`
  - `toLowerCase()`
  - `toUpperCase()`

Universidad de Deusto
ESIDE

Referenciando Objetos en JavaScript

- Para hacer referencia al documento utilizamos `window.document` o `document`
- Dado el siguiente fragmento de código HTML:

```
<body>
  <form name="boxes">
    Enter first name: <input type="text"
  name="fName"><br>
    Enter last name: <input type="text"
  name="lName">
  </form>
</body>
```

- Para hacer referencia a la primera entrada de texto podríamos hacer:

```
document.form.fName
```

- Alternativamente:

```
document.boxes.fName
```

- Ó:

```
document.forms[0].elements[0]
```

Universidad de Deusto
ESIDE

Ejemplo Función JavaScript

```
function trimAll(str) {
  if (str!=null) {
    while (str.length > 0 &&
   “\n\r\t”.indexOf(str.charAt(str.length - 1)) != -1)
      str=str.substring(0, str.length – 1);
    while (str.length > 0 &&
   “\n\r\t”.indexOf(str.charAt(0)) != -1)
      str = str.substring(1, str.length);
  }
  return str;
}
```

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: DOM

- Document Object Model (DOM) es una forma de representar documentos estructurados jerárquicos como un modelo orientado a objetos.
- DOM el el estándar oficial de la World Wide Web Consortium (W3C) para representar documentos estructurados en una manera neutral a la plataforma y al lenguaje de programación.
- DOM fue soportado inicialmente por los navegadores web para manipular elementos en un documento HTML. A través de DOM se puede acceder y actualizar dinámicamente el contenido, estructura y estilo de los documentos. Para evitar las incompatibilidades de las diferentes implementaciones de DOM en los diferentes navegadores, el W3C desarrolló la especificación DOM.
- Un documento DOM bien formado tiene la forma de un árbol. La mayoría de los procesadores de XML (Xerces) o XSL como Xalan han sido desarrollados para hacer uso de esta estructura, donde el documento es parseado completamente y guardado en memoria.

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: DOM

- DOM es un árbol que representa al documento visualizado en un navegador
- La funcionalidad del WC3 DOM hace posible la creación de documentos web extremadamente dinámicos.
- Un documento está conformado por varios nodos, cada uno representado un elemento del documento
- Un árbol establece una relación jerárquica entre elementos
  - Por ejemplo, el documento es un nodo que puede contener nodos imágenes, los cuales pueden contener atributos (relación parent-child)

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: DOM

- Podemos acceder a cualquier elemento de un documento a través de su `ID` y los siguientes métodos:
  - `getElementById()` → devuelve una referencia al nodo correspondiente a un `ID`
    - Por ejemplo:
      ```
      <div id="myElement"></div>
      ```
    - Se puede acceder en JavaScript como:
      ```
      x = document.getElementById("myElement")
      ```
  - `getElementsByTagName()` → devuelve todos los elementos de un documento correspondientes al tipo de etiqueta pasado
    - `x = document.getElementsByTagName("img")` → devolvería todos los elementos imagen en un documento
    - Para recuperar el primer elemento haríamos: `firstImage = x.item(0)`

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: DOM

- Cada elemento en un árbol DOM tiene una serie de métodos que permiten manipularlo
- Dado el siguiente fragmento XHTML:

```
<img id="myImg" src="jabberwocky.gif" align="left"
  />
```

- Podemos recuperar una referencia al mismo mediante:

```
x = document.getElementById("myImg")
```

- Los siguientes métodos serán disponibles a `x`:
  - `getAttribute()` → devuelve el valor del atributo pasado

    `myValue = x.getAttribute("src")` // devuelve jabberwocky.gif
  - `setAttribute()` → asigna un valor a un atributo

    `x.setAttribute("align","right")`
  - `removeAttribute()` → borra el atributo pasado

    `x.removeAttribute("align")`

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: DOM

- Además, podemos aplicar el método `getElementsByTagName()` a un elemento cualquiera del árbol
  - En XHTML:

```
<table id="tableTwo">
  <tr>
    <td>This is Cell Four</td>
    <td>This is Cell Five</td>
    <td>This is Cell Six</td>
  </tr>
</table>
```

  - En Javascript:

```
x = document.getElementById("tableTwo")
tableCells = x.getElementsByTagName("TD")
```

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: DOM

- A través del nodo raíz podemos crear nuevos nodos, destruirlos o moverlos alrededor
  - `createElement` → crea una nueva etiqueta
    ```
    myImg = document.createElement("IMG")
    ```
  - Luego le añadimos atributos :
    ```
    myImg.setAttribute("id","imageOne")
    myImg.setAttribute("src","jabberwocky.gif")
    ```
  - Y lo colocamos en el árbol bajo el elemento body o cualquier otro elemento mediante `appendChild`:
    ```
    docBody =
      document.getElementsByTagName("body").item(0)
    docBody.appendChild(myImg)
    ```
  - Podemos añadir elementos textuales mediante `createTextNode()`

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: DOM

```
function buildTable() {
   docBody = document.getElementsByTagName("body").item(0);
   myTable = document.createElement("TABLE");
   myTable.id ="TableOne“;
   myTable.border = 1;
   myTableBody = document.createElement("TBODY");
   for (i = 0; i < 3; i++) {
      row = document.createElement("TR");
      for (j = 0; j < 3; j++) {
         cell = document.createElement("TD");
         cell.setAttribute("WIDTH","50");
         cell.setAttribute("HEIGHT","50");
         textVal = "Cell" + i + "_" + j;
         textNode = document.createTextNode(textVal);
         cell.appendChild(textNode);
         row.appendChild(cell);
      }
      myTableBody.appendChild(row);
   }
   myTable.appendChild(myTableBody);
   docBody.appendChild(myTable);
}
window.onload = buildTable();
```

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: DOM

- Otros métodos disponibles para navegar a través de DOM son:
  - `replaceChild(new,old)` → reemplaza un nodo por otro

```
newRow = document.createElement("TR");
tableBody = document.getElementById("myTableBody");
oldRow = document.getElementsByTagName("TR").item(0);
tableBody.replaceChild(newRow,oldRow);
```

  - `removeChild(child)` → borra un nodo

```
tableBody = document.getElementById("myTableBody");
childOne =
  document.getElementsByTagName("TR").item(0);
tableBody.removeChild(childOne);
```

  - `insertBefore(new,ref)` → inserta un nuevo nodo delante de otro

```
newRow = document.createElement("TR");
tableBody = document.getElementById("myTableBody");
allRows = tableBody.getElementsByTagName("TR");
lastRow = allRows.item(allRows.length-1);
tableBody.insertBefore(newRow,lastRow) ;
```

  - `hasChildNodes()` → devuelve `true` si un nodo tiene hijos

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: DOM

- Por otro lado cada nodo tiene las siguientes propiedades:
  - `nodeName` → devuelve el nombre de una etiqueta HTML como un string

```
x = document.getElementById("myElement");
if (x.nodeName == "IMG")
    alert("myElement is an image tag");
```

  - `nodeType` → devuelve un entero representando el tipo de un elemento
    - (1) → nodo elemento, (2) → atributo y (3) texto

```
x = document.getElementById("myElement");
if (x.nodeType == 1) {
   alert("myElement is an element node")
}
```

  - `nodeValue` → el valor de un nodo

```
a = document.getElementById("myElement");
b = a.childNodes.item(4).nodeValue
```

  - `parentNode` → devuelve el nodo padre del actual

```
x = document.getElementById("myElement");
xParent = x.parentNode ;
```

  - `childNodes`→ devuelve los hijos de un nodo

```
a = document.getElementById("myElement");
b = a.childNodes.length;
b = a.childNodes.item(2);
```

  - `firstChild` y `lastChild` apuntan directamente al primer y último elemento en la colección childNodes

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: XSLT

- Los navegadores actuales son capaces de realizar transformaciones XSLT
- La instrucción de procesamiento `xml-stylesheet` indica cómo realizar tal transformación
- Consideremos el documento XML:

```
<?xml-stylesheet href="squareAsHTML.xsl"
    type="text/xsl" ?>
<numbers>
  <number>2</number>
  <number>11</number>
  <number>100</number>
  <number>-5</number>
</numbers>
```

Universidad de Deusto
ESIDE

Manipulando XML en AJAX: XSLT

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
<!-- squareAsHTMLJS1.xml: create an HTML document with JavaScript that
     interactively computes the square of each "number" element read from
     the source tree. -->

  <xsl:template match="/"> <!-- Set up web page -->
    <html>
      <head>
        <title>Squares</title>
        <script language="JavaScript1.2">
          function showSquare(n) {
           alert("the square is " + n*n);
          }
        </script>
        <style> <!-- Put a little CSS in -->
           body { font-family: arial,helvetica; }
           h1   { font-size: 14pt }
           p    { font-size: 10pt}
        </style>
      </head>
      <body>
        <h1>Squares</h1>
        <p>Click a button to see the square of that number.</p>
        <form>
        <xsl:apply-templates/>
        </form>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="number">
    <p><input type="button" value=" {.} " onClick="showSquare({.})"/></p>
  </xsl:template>

</xsl:stylesheet>
```

Universidad de Deusto
ESIDE

Documentación XSLT Processor

- Para acceder desde JavaScript a XSLT revisar documentación en:
  - http://www.mozilla.org/projects/xslt/js-interface.html

Universidad de Deusto
ESIDE

La magia de AJAX: El objeto `XMLHttpRequest`

- Gracias a este objeto los clientes web pueden recuperar y entregar XML, en background → **ASÍNCRONAMENTE**.
- Usando DOM podemos leer el contenido devuelto y modificar el documento presente en el navegador

Universidad de Deusto
ESIDE

Creación del XMLHttpRequest

- Desafortunadamente en su creación hay que tener en cuenta las diferencias entre navegadores:
- En Mozilla y Safari se hará:

```
var req = new XMLHttpRequest();
```

- En el ActiveX de Internet Explorer haríamos:

```
var req = new
  ActiveXObject("Microsoft.XMLHTTP");
```

- ...o en las últimas versiones de IE:

```
xmlreq = new ActiveXObject("Msxml2.XMLHTTP");
```

- Este objeto puede trabajar de manera asíncrona al usuario

Universidad de Deusto
ESIDE

Métodos del XMLHttpRequest

- `abort()` → para la invocación actual
- `getAllResponseHeaders()` → devuelve todas las cabeceras (nombre-valor) como un string
- `getResponseHeader("headerLabel")` → devuelve el valor asociado a una cabecera
- `open("method", "URL"[, asyncFlag[, "userName"[, "password"]]])` → abre una conexión remota
  - Usar POST para invocaciones que envían datos al servidor, de un tamaño superior a 512 bytes
  - Usar GET para recuperar datos del servidor
  - Si `asyncFlag` es '`true`' hay que definir un método que recibirá un callback cuando se genere el evento `onreadystatechange`
- `send(content)` → envía la petición con un contenido opcional
- `setRequestHeader("label", "value")` → añade una cabecera a la petición

Ejemplo XMLHttpRequest

```
var req = null;
function loadXMLDoc(url) {
  // branch for native XMLHttpRequest object
  if (window.XMLHttpRequest) {
    req = new XMLHttpRequest();
  // branch for IE/Windows ActiveX version
  } else if (window.ActiveXObject) {
    // Create XMLHttpRequest via MS ActiveX
    try { // Try to create XMLHttpRequest in later IE versions
      xmlreq = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e1) { // Failed to create required ActiveXObject
      try { // Try version supported by older IE versions
        xmlreq = new ActiveXObject("Microsoft.XMLHTTP");
      } catch (e2) {
        // Unable to create an XMLHttpRequest with ActiveX
      }
    }
  }
  if (req) {
    req.onreadystatechange = processReqChange;
    req.open("GET", url, true);
    req.send(null);
  }
}
```

Universidad de Deusto
ESIDE

Propiedades Métodos del XMLHttpRequest

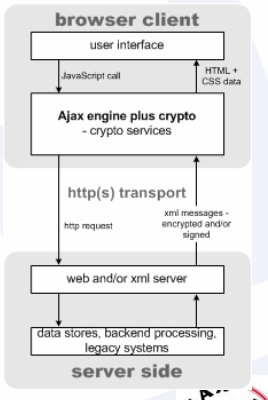
- Propiedades del objeto `XMLHttpRequest`:
  - `onreadystatechange` → asocia el manejador de eventos invocado cada vez que cambie el estado de una petición
  - `readyState` → entero con estado de objeto:
    - 0 = no inicializado
    - 1 = cargando
    - 2 = cargado
    - 3 = interactivo
    - 4 = completado
  - `responseText` → contenido textual de la respuesta enviada por el servidor
  - `responseXML` → documento DOM correspondiente a la respuesta del servidor
  - `status` → código numérico devuelto por el servidor
  - `statusText` → mensaje de texto acompañando al código de status

Ejemplo procesamiento eventos XMLHttpRequest

```
function processReqChange() {
    // only if req shows "loaded"
    if (req.readyState == 4) {
        // only if "OK"
        if (req.status == 200) {
            // ...processing statements go here...
        } else {
            alert("There was a problem retrieving the
  XML data:\n" +
                req.statusText);
        }
    }
}
```

Seguridad en AJAX

- Por motivos de seguridad las invocaciones a recursos vía HTTP solamente pueden efectuarse a páginas alojadas en el mismo dominio que el de la página que contenía el script
- Si queremos que la información de nuestras peticiones no sea visible deberemos utilizar un canal HTTPS.



Universidad de Deusto
ESIDE

DTML Document Object Model

- Da acceso a todo elemento en un documento
- Para acceder un elemento dentro de un documento, éste ha de tener un identificador

```
<html>
  <body>
    <h1 id="header">My header</h1>
    <script type="text/javascript">
document.getElementById('header').style.color="red"
    </script>
  </body>
</html>
```

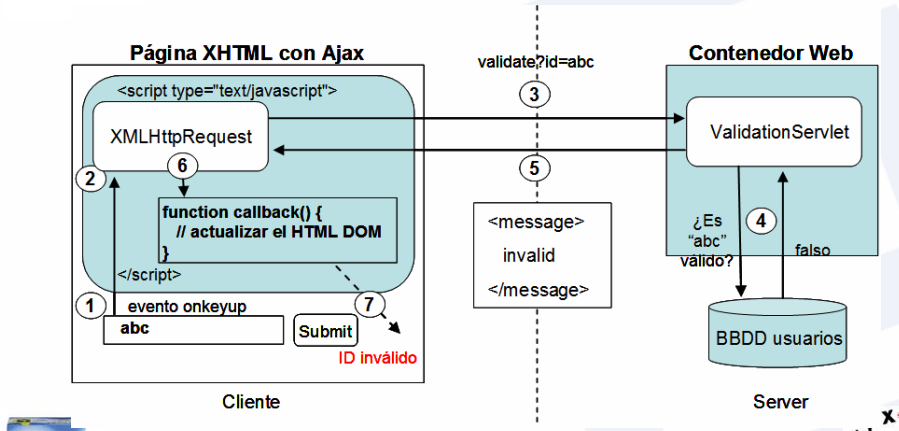
Universidad de Deusto
ESIDE

Nuestra primera aplicación AJAX

- Vamos a considerar una aplicación web con un formulario que requiere validación en el servidor sin necesidad de refrescar la página completamente
  - Es decir, realizar un HTTP POST de la página implícitamente
- La figura en la siguiente transparencia ilustra el funcionamiento de esta aplicación

Universidad de Deusto
ESIDE

Nuestra primera aplicación AJAX



Universidad de Deusto
ESIDE

Nuestra primera aplicación AJAX

- Los flujos de control serían:
  1. Un evento ocurre
  2. Un objeto `XMLHttpRequest` es creado e inicializado
  3. Se realiza una invocación remota a través del objeto `XMLHttpRequest`
  4. La petición es procesada por el componente servidor `ValidationServlet`
  5. Este servlet devuelve un documento XML con el resultado
  6. El objeto `XMLHttpRequest` invoca la función `callback()` y procesa el resultado
  7. El árbol DOM de la página web es actualizado

Un evento ocurre

- Como resultados de eventos generados por la interacción del usuario funciones JavaScript son invocadas
- Ejemplo:

```
<input type="text“
       size="20“
       id="userID“
       name="id“
       onkeyup="validate();">
```

El objeto `XMLHttpRequest` es creado

```
var req;
function validate() {
   var idField = document.getElementById("userID");
   var url = "validate?id=" + escape(idField.value);
   if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
        req = new ActiveXObject("Microsoft.XMLHTTP");
   }
   req.open("GET", url, true);
   req.onreadystatechange = callback;
   req.send(null);
}
```

- Los tres parámetros del método `open()` del objeto `XMLHttpRequest` indican el método HTTP utilizado, la url de destino y si el método se invocará de manera asíncrona o no
- `req.onreadystatechange = callback;` indica la función a invocar cuando la petición sea resuelta

Universidad de Deusto
ESIDE

El objeto `XMLHttpRequest` realiza una invocación remota

- `req.send(null);` realiza la invocación, se pasa `null` porque un HTTP GET no tiene cuerpo
- Si se quiere utilizar un HTTP POST entonces:
  - Deberíamos enviar la cabecera `Content-Type`
- Ejemplo HTTP POST:

```
req.setRequestHeader("Content-Type",
  "application/x-www-form-urlencoded");
req.send("id=" +
  escape(idTextField.value));
```

Universidad de Deusto
ESIDE

La petición es procesada por `ValidationServlet`

```
// compile: javac -classpath %TOMCAT_HOME%\common\lib\servlet-api.jar ValidationServlet.java
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ValidationServlet extends HttpServlet {
    private ServletContext context;
    private HashMap users = new HashMap();

    public void init(ServletConfig config) throws ServletException {
        this.context = config.getServletContext();
        users.put("solop","account data");
        users.put("solop","account data");
    }

    public void doGet(HttpServletRequest request, HttpServletResponse  response)
        throws IOException, ServletException {
        String targetId = request.getParameter("id");
        if ((targetId != null) && !users.containsKey(targetId.trim())) {
            response.setContentType("application/xml");
            response.setHeader("Cache-Control", "no-cache");
            response.getWriter().write("<message>invalid</message>");
        } else {
            response.setContentType("text/xml");
            response.setHeader("Cache-Control", "no-cache");
            response.getWriter().write("<message>valid</message>");
        }
    }
}
```

Universidad de Deusto
ESIDE

La petición es procesada por `ValidationServlet`

- La respuesta necesariamente tiene que ser en XML, para evitar caching de respuestas se utiliza la cabecera `Cache-Control`:

```
response.setContentType("text/xml");
response.setHeader("Cache-Control",
  "no-cache");
response.getWriter().write("<message
  >valid</message>");
```

Universidad de Deusto
ESIDE

El objeto `XMLHttpRequest` invoca la función `callback` y procesa su resultado

- La función `callback` es invocada cuando se producen cambios en el `readyState` del objeto `XMLHttpRequest`. Si `readyState == 4` significa que la invocación ha acabado:
- Una vez recibida la respuesta la representación DOM de un documento XHTML puede ser modificado incluyendo el contenido de la respuesta.
  - `req.responseText` → devuelve el contenido textual de la respuesta
  - `req.responseXML` → la respuesta XML a la que se le pueden aplicar métodos del estándar DOM

El objeto `XMLHttpRequest` invoca la función `callback` y procesa su resultado

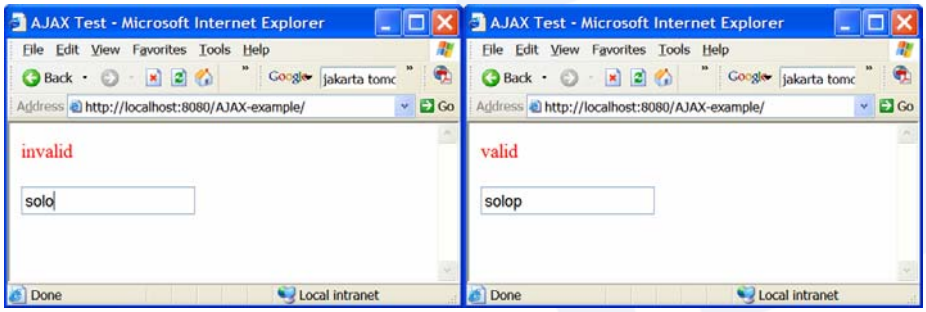
```
function callback() {
  if (req.readyState == 4) {
    if (req.status == 200) {
      // update the HTML DOM based on whether or not message is valid
      parseMessage();
    }
  }
}

function parseMessage() {
  var message = req.responseXML.getElementsByTagName("message")[0];
  setMessage(message.childNodes[0].nodeValue);
}

function setMessage(message) {
  var userMessageElement = document.getElementById("userIdMessage");
  userMessageElement.innerHTML = "<font color=\"red\">" + message + "
   </font>";
}
```

Universidad de Deusto
ESIDE

El XHTML DOM es actualizado

- JavaScript puede recoger una referencia a un elemento en el HTML DOM
- Por ejemplo, a través de la invocación `document.getElementById("userIdMessage")`
- Con una referencia a un elemento:
  - Añadir, modificar o eliminar atributos
  - Añadir, modificar o eliminar elementos

Aplicación en Ejecución



Problemas de AJAX

- **Disponibilidad del objecto `XMLHttpRequest`**
  - Aunque la mayoría de las últimas versiones soportan AJAX hay versiones antiguas y algunos navegadores que no lo hacen
  - Usar `XMLHttpRequest` puede provocar que algunos usuarios no puedan ver tu web
  - Por tanto, hay que proveer un mecanismo alternativo para asegurarse que todo el mundo pueda ver tu portal AJAX
- **Usabilidad**
  - Es necesario avisar en ocasiones que una acción lanzada asíncronamente está en curso
    - Por ejemplo, mensajito diciendo "Actualizándose ..."
    - Cambiar colores de la parte de la página cambiada
  - Problemas con que el botón de "Atrás" del navegador no funciona y la URL mostrada no representa el estado actual de la página

Universidad de Deusto
ESIDE

Problemas de AJAX

- **Carga del Servidor**
  - Crear una interfaz de usuario en vez de un form normal puede incrementar el número de peticiones que llegan al servidor
    - Google Suggest
  - Hay que tener en cuenta la sobrecarga que puede ser introducida al usar Ajax
    - Hay que bufear peticiones y cachear respuestas en el cliente
    - Delegar todo lo que se pueda a la parte cliente
- **Comportamiento Asíncrono**
  - Las peticiones `XMLHttpRequest` puede que no concluyan en el orden en que fueron emitidas
    - Se puede incluir un timestamp junto con cada respuesta

Universidad de Deusto
ESIDE

Refactoring JavaScript

- JavaScript ha pasado de ser un lenguaje utilizado para aplicar pequeñas mejoras a una web, a servir para el desarrollo de completas y complejas aplicaciones
- Refactoring es una técnica para mejorar su estructura lógica interna y preservar su funcionalidad
- Buena guía en: http://www.softwaresecretweapons.com/jspwiki/Wiki.jsp?page=JavascriptRefactoringForSaferFasterBetterAJAX

Ejemplo de mal JavaScript

```
var req;
var ready;
function loadXMLDoc(url) {
    // this function uses req, ready,
  processStateChange()
}
function processStateChange() {
    // this function uses uses req, ready,
  processXML()
}
function processXML() {
}
```

Universidad de Deusto
ESIDE

Reglas de diseño general

- Evitar el uso de variables globales
- Permitir que varias peticiones `XMLHttpRequest` se ejecuten concurrentemente
- Encapsular código de navegador específico
- Mejorar el manejo de errores y su presentación
- Crear archivos JavaScript reutilizables en otros proyectos

Universidad de Deusto
ESIDE

1. Colocar las funciones y variables globales en un objeto

```
function oyXMLRPCProvider(){
    var req;
    var ready;

    var loadXMLDoc = function (url) {
        // this function uses req, ready, processStateChange()
    }

    var processStateChange = function () {
        // this function uses uses req, ready, processXML()
    }

    var processXML = function () {

    }
}
```

Universidad de Deusto
ESIDE

2. Declarar todas las variables como privadas y clasificar funciones

```
function oyXMLRPCProvider(){
    var req;
    var ready;

    this.loadXMLDoc = function (url) {
        // this function uses req, ready, processStateChange()
    }

    var processStateChange = function () {
        // this function uses uses req, ready, onComplete()
    }

    this.onComplete = function () {
    }

}
```

Universidad de Deusto
ESIDE

3. Hacer logging de información y mejorar la gestión de errores

```
function oyXMLRPCProvider() {
    // place here all code from previous step
    var oThis = this;

    // checks to see if we have too many messages in log
    var internalCanMsg = function(){
        msgCount++;
        return msgCount < 100;
    }

    // adds message to internal log
    var internalOnLog = function(msg){
        if(oThis.onLog && internalCanMsg()) {
            oThis.onLog(msg);
        }
    }

    // adds message to internal error handler
    var internalOnError = function(msg){
        if(oThis.onError && internalCanMsg()) {
            oThis.onError(msg);
        }
    }

    this.onLog = function (msg){
    }

    this.onError = function (msg){
    }
}
```

Universidad de Deusto
ESIDE

4. Uso de la clase generada

```
function demo(){
  // clear log
  document.getElementById('output').value = "";
  // create logging function

  var myOnLog = function(msg){
    document.getElementById('output').value += msg + "\n";
  }

  // create completion function
  var myOnComplete = function(responseText, responseXML){
    document.getElementById('outputText').value = responseText;

    alert("Done!");
  }

  // create provider instance; wire events
  var provider = new oyXMLRPCProvider();
  provider.onComplete = myOnComplete;
  provider.onLog = myOnLog;
  provider.onError = myOnLog;
  provider. loadXMLDoc ("../data/data.xml");
}
```

Universidad de Deusto
ESIDE

Frameworks AJAX

- Las frameworks JavaScript basadas en la parte cliente se dividen en:
  - Frameworks de Infraestructura, abstraen al desarrollador de las ideosincracias de los diferentes navegadores
    - AjaxJS, GoogleAJAXSLT, Sarissa
  - Frameworks de aplicación, aparte de encapsular el funcionamiento de `XmlHttpRequest` también incluyen widgets al estilo de las GUI frameworks
    - Tibet, DOJO
- Los tipos de frameworks en la parte servidora son:
  - Generación de HTML/JS se realiza automáticamente en el servidor
  - Invocación remota, se crea un túnel directamente desde JavaScript a funciones en la parte servidora
    - Direct Web Remoting (JAVA), AjaxAC (PHP) y Ajax.NET
- Las frameworks más conocidas aparecen listadas en http://www.ajaxpatterns.org/AJAXFrameworks
- Microsoft quiere incluir soporte AJAX en ASP.NET 2.0 con Atlas: http://www.asp.net/default.aspx?tabindex=7&tabid=47

Universidad de Deusto
ESIDE

Taxonomía Frameworks AJAX

- **Frameworks JavaScript puras**:
  - **Frameworks de infraestructura** → proveen código independiente del browser
    - Wrapper alrededor de `XMLHttpRequest` (todas las frameworks lo ofrecen)
    - Manipulación e interrogación de XML
    - Manipulación del DOM HTML a partir de las respuestas de `XMLHttpRequest`
  - **Frameworks de aplicación**
    - Proveen lo mismo que las anteriores frameworks
    - Además ofrecen widgets de manera similar a frameworks de GUI
- **Basadas en la parte servidora**

Taxonomía Frameworks AJAX

- **Basadas en la parte servidora**
  - **Generación de HTML/JS**: el servidor provee la generación completa de código y la coordinación entre el navegador y el servidor
  - **Invocación remota**: las llamadas JavaScript son ruteadas directamente a funciones en la parte servidora y las respuestas a callbacks JavaScript.

Frameworks JavaScript Puras: Aplicación

- DOJO (http://dojotoolkit.org) ofrece soporte de widgets y de mensajería entre navegador y servidor
  - Framework para crear widgets JavaScript propietarios
  - Librería de widgets preconstruidos
  - Soporte para mensajería browser-servidor
  - Soporte para manipular urls en el navegador
  - Es de código abierto
- Open Rico (http://openrico.org/demos.page) es otra posibilidad

Ejemplo DOJO – Cliente

```
<!-- This is an attempt to create the simplest possible Dojo example. -->
<html>
<head>
<script language="JavaScript" type="text/javascript“ src="dojo.js"></script>
<script language="JavaScript" type="text/javascript">
function validate(id) {
   var value = document.getElementById(id).value;
   dojo.io.bind({
      url: "validate.jsp“,
      content: { username: value },
      handler: function(type, data, evt){
      alert(data); },
      mimetype: "text/plain"
    });
}
</script>
</head>
<body>
    <p>Username “dipina" is taken; any other value should return "ok".</p>
    <input type="text" id="username" name="username" size="20" value=""><button
    onclick="validate('username')">Validate</button>
</body>
</html>
```

Universidad de Deusto
ESIDE

Ejemplo DOJO – Servidor

```
<%
   if (request.getParameter("username") != null &&
   request.getParameter("username").toLowerCase().equa
   ls("dipina")) {
       response.getWriter().write("taken");
   } else {
       response.getWriter().write("available");
   }

%>
```

Universidad de Deusto
ESIDE

Frameworks JavaScript Puras: Infraestructura

- **Sarissa** (http://sarissa.sf.net) es una API JavaScript que encapsula la funcionalidad XML en llamadas independientes del navegador
  - Creación del objeto `XMLHttpRequest` portable
  - Consultas XPath portables
  - Manipulación DOM portable
  - XSLT portable
  - Serialización a XML portable
  - De código abierto
- Artículo sobre Sarissa en: http://www.xml.com/lpt/a/2005/02/23/sarissa.html
- Documentación de Sarissa en http://sarissa.sourceforge.net/doc/
- Otras frameworks alternativas son: libXmlRequest e Interactive Website Framework

Universidad de Deusto
ESIDE

Usando Sarissa

- Para obtener un objeto DOM en Sarissa haríamos:

```
var oDomDoc = Sarissa.getDomDocument();
```

- Para obtener el objeto `XMLHttpRequest` haríamos de manera síncrona:

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET",
  "http://foo.org/someDocument.xml", false);
xmlhttp.send(null);
alert(Sarissa.serialize(xmlhttp.responseXML));
```

- Y asíncrona:

```
xmlhttp.onreadystatechange = function() {
  if(xmlhttp.readyState == 4)
    alert("Finished loading!");
}
```

Usando Sarissa

- Cargando un documento remoto:

```
var oDomDoc = Sarissa.getDomDocument();
oDomDoc.async = true; // this line can be ommited - true is the
   default
// we add a listener for the load event to know when loading is
   finished
function myHandler {
   if(oDomDoc.readyState == 4)
    alert(Sarissa.serialize(oDomDoc));// show the file contents
};
oDomDoc.onreadystatechange = myHandler;
oDomDoc.load("someDocument.xml");
```

- Creando un documento XML de un string:

```
var oDomDoc = Sarissa.getDomDocument();
var xmlString = "<root>my xml!</root>";
oDomDoc = (new DOMParser()).parseFromString(xmlString, "text/xml");
alert(Sarissa.serialize(oDomDoc));
```

Universidad de Deusto
ESIDE

Usando Sarissa

- Como serializar un objeto no XML:

```
// create an object hierarchy
book.chapters = new Array();
book.chapters[0] = "Kingdom of Tags";
book.chapters[1] = "Fall";
book.chapters[2] = "Final battle";
book.chapters[3] = "Characters that need to be escaped: << << \"'
   \"\"\"&&'' < > & ' \" ";
book.chapters[4] = "Epilogue";
book.editor = "Manos Batsis";
var publisher = new Object();
publisher.name = "Some Publisher";
book.publisher = publisher;

// serialize to an XML string
var s = Sarissa.xmlize(book, "book");
alert("Generated XML:\n"+s)
```

- También permite hacer transformaciones XSLT y XPATH independientes del procesador (ejemplo Sarissa)

Universidad de Deusto
ESIDE

Frameworks JavaScript: Parte Servidora Multilenguaje

- **SAJAX** (http://www.modernmethod.com/sajax/) rutea las llamadas directamente de JavaScript en tu lenguaje de back-end y viceversa
  - Puedes llamar al método JavaScript "x_calculateBudget()" e invocará en el servidor el método Java calculateBudget() que devolverá el valor al método javascript x_calculateBudget_cb().
  - Facilita el mapeo de JavaScript a las operaciones del back-end
  - Capaz de mapear las llamadas a diferentes plataformas servidoras: ASP/ColdFusion/Io/Lua/Perl/PHP/Python/Ruby.
  - Open-source
- **Ajax.Net** (http://ajax.schwarz-interactive.de/csharpsample/default.aspx) es una librería que permite el acceso desde JavaScript a código servidor en .NET
  - Como SAJAX gestiona la invocación de peticiones en la parte servidora y la comunicación de respuestas en la parte cliente
  - Puede acceder datos de sesión en JavaScript
  - Cachea resultados
  - Libre, código disponible, sin licencia especificada
  - Existe un excelente tutorial en: http://www.developerfusion.co.uk/show/4704/

Universidad de Deusto
ESIDE

Ejemplo SAJAX-PHP

```
<?
require("Sajax.php");
function multiply($x, $y) {
  return $x * $y;
}
sajax_init();
sajax_export("multiply");
sajax_handle_client_request();
?>
<html><head><title>Multiplier</title>
  <script>
    <? sajax_show_javascript();      ?>
    function do_multiply_cb(z) {
      document.getElementById("z").value = z;
    }
```

Universidad de Deusto
ESIDE

Ejemplo SAJAX-PHP

```
  function do_multiply() {
    var x, y;
    x = document.getElementById("x").value;
    y = document.getElementById("y").value;
    x_multiply(x, y, do_multiply_cb);
  }
</script>
</head>
<body>
  <input type="text" name="x" id="x" value="2" size="3">*
  <input type="text" name="y" id="y" value="3" size="3">=
  <input type="text" name="z" id="z" value="" size="3">
  <input type="button" name="check" value="Calculate"
      onclick="do_multiply(); return false;">
</body>
</html>
```

Universidad de Deusto
ESIDE

Configurando WAMP

- Intalar Apache 2 a partir del fichero .msi de la carpeta downloads
- Instalar PHP5 a partir del instalador incluido en downloads
- Tras instalar Apache 2, modificar fichero `httpd.conf` introduciendo las siguiente líneas:

```
LoadModule php5_module "c:/php/php5apache2.dll"
AddType application/x-httpd-php .php
# configure the path to php.ini
PHPIniDir "C:/php "
```

- Añadir a la variable de entorno PATH, el directorio donde se ha instalado PHP5
- En `php.ini` descomentar la línea correspondiente a la extensión PHP para MySQL

```
extension=php_mysql.dll
```

- Copiar `libmysql.dll` a `c:\windows\system32`

Universidad de Deusto
ESIDE

Frameworks JavaScript Puras: Parte Servidora Java

- **Direct Web Remoting** (http://www.getahead.ltd.uk/dwr/) es una framework para invocar a métodos Java desde JavaScript
  - Similar a SAJAX pasa invocaciones de JavaScript a métodos Java y devuelve las repuestas invocando callbacks JavaScript
  - Puede utilizarse como una framework web - Struts, Tapestry, etc.
  - Sigue la filosofía de Spring (KISS/POJO/orthogonality).
  - Open-source
- **Echo 2** (http://www.nextapp.com/products/echo2/) permite codigicar aplicaciones Ajax en Java
  - Tiene una demo muy interesante en http://demo.nextapp.com/InteractiveTest/ia
  - Genera automáticamente HTML y Javascript.
  - Co-ordina los mensajes XML entre navegador y servidor
  - Permite la creación de controles JavaScript
  - Open source

Universidad de Deusto
ESIDE

Frameworks JavaScript Puras: Parte Servidora PHP

- **AjaxAC** (http://ajax.zervaas.com.au/) encapsula la aplicación en una clase PHP
  - Todo el código de la aplicación es autocontenido en una clase que hace referencia al JavaScript y a la funcionalidad para manejar eventos JavaScript
  - Infraestructura para crear peticiones y manejarlas
  - Todos los eventos JavaScript son asignados dinámicamente
  - Facilmente integrable con motores de templates
  - Estructura de widgets extensible

Universidad de Deusto
ESIDE

Google Maps API

- Disponible en la siguiente url:
  - http://www.google.com/apis/maps/
- Es preciso registrarse y obtener una clave de acceso

Universidad de Deusto
ESIDE

Ejemplo de Acceso a Google Maps

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Google Maps JavaScript API Example - simple</title>
    <script src=http://maps.google.com/maps?file=api&v=1&key=abcdefg
   type="text/javascript"></script>
  </head>
  <body>
    <div id="map" style="width: 300px; height: 300px"></div>
    <script type="text/javascript">
    //<![CDATA[

    if (GBrowserIsCompatible()) {
      var map = new GMap(document.getElementById("map"));
      map.centerAndZoom(new GPoint(-122.141944, 37.441944), 4);
    }
    //]]>
    </script>
  </body>
</html>
```

Universidad de Deusto
ESIDE

Nuevas Aplicaciones AJAX

- Equivalente web de Microsoft Office:
  - http://online.thinkfree.com – online office suite
  - Writely
- Meebo.com – Instant messaging basado en AJAX
- http://www.zimbra.com/ -- Groupware
- Project Management -- http://www.basecamphq.com/
- Gestión de citas, calendario -- http://www.calendarhub.com/
- Openomy – online file system -- http://www.iseff.com/2005/07/introducing-openomy.html

Universidad de Deusto
ESIDE

Desarrollando un lector RSS en AJAX

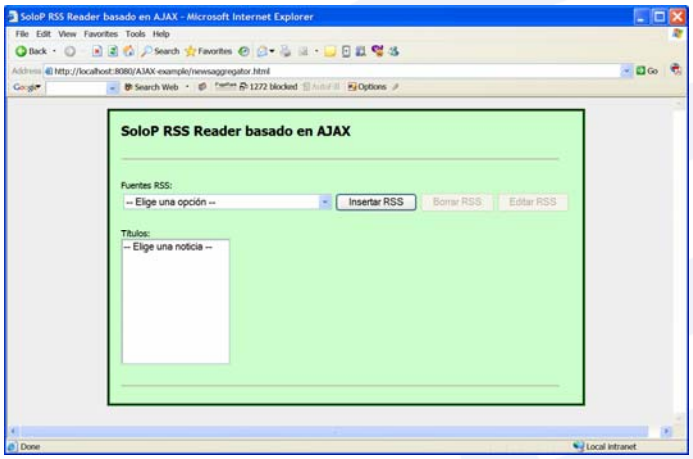
- Vamos a definir un lector de noticias RSS, capaz de gestionar las fuentes RSS
- RSS es una familia de ficheros XML para la sindicación web usando webs de noticas y weblogs.
- Su nombre hace referencia a uno de los siguientes estándares:
  - Rich Site Summary (RSS 0.91)
  - RDF Site Summary (RSS 0.9, 1.0 and 1.1)
  - Really Simple Syndication (RSS 2.0)
- RSS ofrece descripciones breves de contenido web junto con enlaces a las versiones completas del documento.
- El fichero XML en el que se envía esta información es denominado RSS feed, webfeed, RSS stream, o RSS channel.
- Gracias a RSS y con la ayuda de un news aggregator podemos monitorizar los cambios en el contendio de un portal que ofrece sindicación RSS.
  - http://blogs.law.harvard.edu/tech/rss#ltenclosuregtSubelementOfLtitemgt

Ejemplo fichero RSS 2.0

```
<rss version="2.0">
  <channel>
    <title>Ejemplo de canal</title>
    <link>http://example.com/</link>
    <description>Ejemplo de fuente RSS</description>
    <language>es</language>
    <item>
      <title>1 &lt; 2</title>
      <link>http://example.com/1_less_than_2.html</link>
      <description>1 &lt; 2, 3 &lt; 4. En HTML, &lt;b&gt; comienza una
    frase en negrita
        y puedes comenzar un enlace con &lt;a href=
      </description>
      <enclosure url="http://rss.org/mp3s/news1.mp3" length="12216320"
    type="audio/mpeg" />
    </item>
  </channel>
</rss>
```

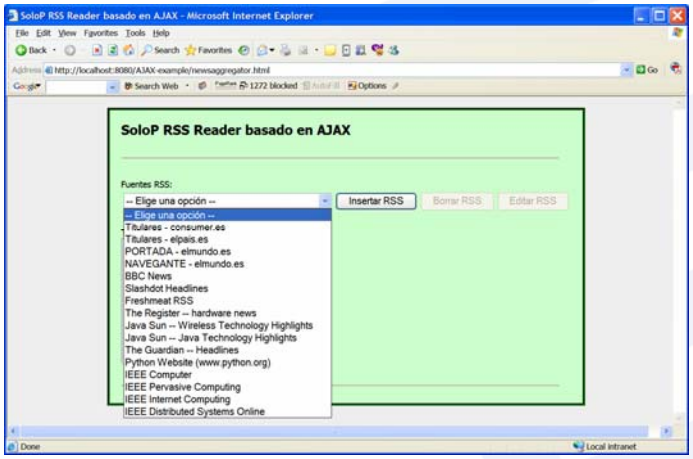
Universidad de Deusto
ESIDE

Desarrollando un lector RSS en AJAX



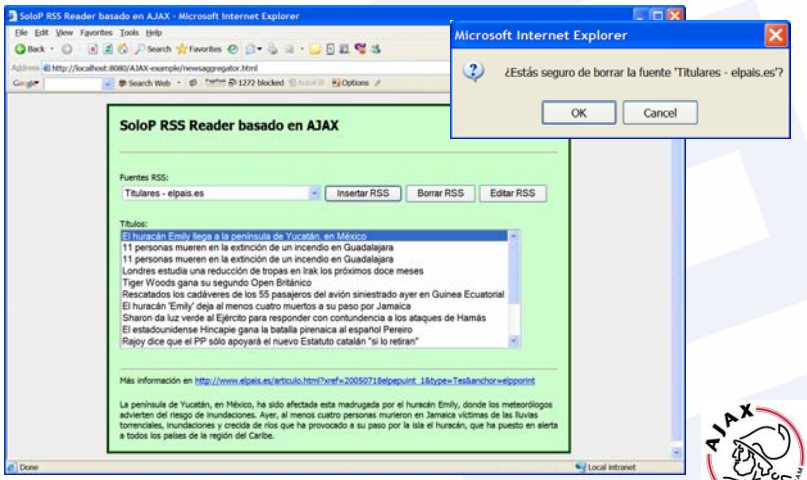
Universidad de Deusto
ESIDE

Desarrollando un lector RSS en AJAX



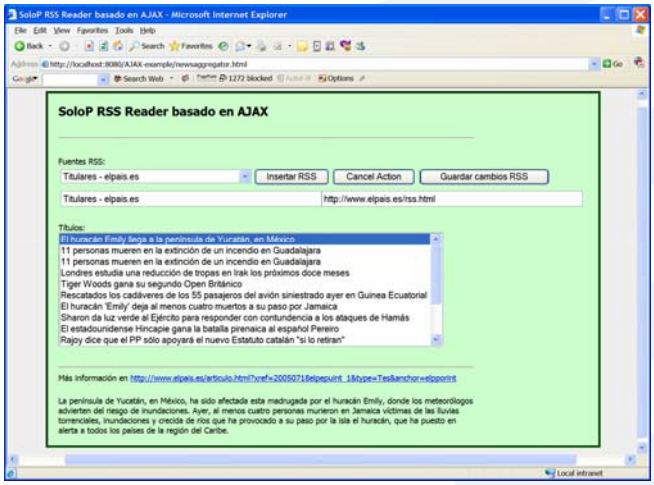
Universidad de Deusto
ESIDE

Desarrollando un lector RSS en AJAX



Universidad de Deusto
ESIDE

Desarrollando un lector RSS en AJAX



Universidad de Deusto
ESIDE

Ejemplo Carrito de la Compra

- Revisar ejemplo `examples\ShoppingCart`, compuesto por 2 paquetes, 1 jsp y 2 js:
  - Paquete `developerworks.ajax.store`, contiene las clases `Cart`, `Catalog` e `Item`
  - Paquete `developerworks.ajax.servlet`, contiene el servlet CartServlet
  - `index.jsp` – visualiza la página inicial
  - `ajax1.js` – contiene las funciones generales mostradas en el artículo
  - `cart.js` – funciones específicas al ejemplo del carrito

Universidad de Deusto
ESIDE

Conclusión

- AJAX es un nuevo enfoque de desarrollo web que garantiza una mayor interactividad y usabilidad de portales web
- Su utilización por líderes en la industria web como Google o Yahoo, da una idea de su potencial.
  - Incluso MicroSoft ya está pensando en añadir a ASP.NET 2.0 un conjunto de controles basados en tecnología AJAX.
- El desarrollo web con AJAX no es trivial y difícil de hacer debugging
  - Sin embargo, las frameworks que están emergiendo simplificarán el desarrollo web basado en AJAX

Universidad de Deusto
ESIDE

Referencias

- XMLHttpRequest –http://developer.apple.com/internet/webcontent/xmlhttpreq.html
- Asynchronous JavaScript Technology and XML (AJAX) With Java 2 Platform, Enterprise Edition
  http://java.sun.com/developer/technicalArticles/J2EE/AJAX/
- HTML DOM – http://www.w3schools.com/htmldom/default.asp
- Javascript – http://en.wikipedia.org/wiki/Javascript
- Core JavaScript Reference – http://research.nihonsoft.org/javascript/CoreReferenceJS15/index.html
- Mi página web – Curso AJAX –, http://paginaspersonales.deusto.es/dipina
- Ajax for Java developers: Build dynamic Java applications – http://www-128.ibm.com/developerworks/library/j-ajax1/?ca=dgr-lnxw01Ajax
- Web 2.0 http://www.web2con.com/
- JavaScript XSLT -- http://www.soi.city.ac.uk/~sa386/epterm2/sqlxml/week9/The%20XSLT-JavaScript%20Interface%20In%20Gecko.htm
- AJAX gives software a fresh look – http://news.com.com/AJAX+gives+software+a+fresh+look/2100-1007_3-5886709.html?tag=st.prev